VBA: Build Simple Calibrated Routes

Contributed by Bert Granberg
15, Jul. 2009
Last Updated 15, Jul. 2009

This code automates the building and calibration of polyline-m route features from street centerlines that carry a route and route-part identifier. In addition to an attributed street centerline input dataset, a feature class of calibration points is required with one calibration point per route part.

Before starting, each route's component centerline features must be coded into separate parts wherever the route is disjoint or broken and wherever spurs and loops occur.

For example, I-80 in Utah consists of two parts, one west of I-15 and one east of I-15. There is a gap of about 3 miles between these parts and the mileposting stops and resumes one either side of this gap. All of the centerlines west of I-15 in the positive direction should carry a RT_NAME = 0080P and a RT_PART = 1, Everything east of I-15 in the positive direction should carry RT_NAME = 0080P and a RT_PART =2. There need to be four calibration points one for each start and end point on each of the 2 route parts.

Looping is always a little problematic and my favored way to represent loops is to put a very small part at the end of the loop (where the feature comes back together) and set its calibration measures to -9999. This breaks the loop into two parts so that none of the start and end points are coincident for a single part. After the route is calibrated this little part can be deleted and you can can reclose the loop.

Route building can of course be done in model builder and the ArcGIS command line editor as well. However, I prefer using this code for large batch route building jobs. It is fast relative to the GP environment of the alternatives.

Important notes:
- This code will attempt to build all the routes for which you have calibration points. If you want to build just a select set of routes, set a definition query (temporary subset) in your calibration layer's properties.
- The route output feature class will carry a date/time stamp in the feature class name.
- Where route have multiple parts (like I-80 described above), you'll likely want to open an edit session and merge the built, calibrated route parts together into a single route feature. With branching and looping the resulting feature may be stored as fewer parts than you start with (some parts will be be merged at spur intersections) but the route and m coordinates should keep their overall integrity.
- Occasionally routes directional orientation will be the reverse of the measure coordinates. This is not a problem within ArcMap unless you want to draw arrowhead symbols on the route features, but this can cause violate some other systems integrity rules. In this case, routes can be flipped within an ArcMap edit session (right click on edit sketch and choose 'Flip.' M coordinates will stay where they are supposed to be.

The code consists of one procedure, BuildRoutesParts_SimpleCalibration(), and 5 related functions. Make sure to investigate and set the parameters required in the BuildRoutesParts_SimpleCalibration procedure.

Good luck.

```
Option Explicit
Public Sub BuildRoutesParts_SimpleCalibration()

    Dim outPath As String
    Dim calibrationLayerIndex As Integer
    Dim roadsLayerIndex As Integer
    Dim pCalibrationValueFieldName As String
    Dim calPtSearchTol As Double
    Dim rtNamepartStr As String

    '**************************
    '***** SET THESE PARAMETERS

    'the file geodatabase where the new route/polyline-m feature class will be created
    outPath = "c:/UDOTLRS/LRSBert.gdb"

    'the position of the CALIBRATION POINT layer in the MXD, zero is the first layer
    calibrationLayerIndex = 0

    'the position of the ROAD CENTERLINE layer in the MXD, zero is the first layer
```

```
    roadsLayerIndex = 1

    'The field name containing the measure value in the CALIBRATION LAYER
    pCalibrationValueFieldName = "REF_VALUE"

    'The field name containing the full route-part name value in the CALIBRATION LAYER
    'Values in this fields must match the format:
    'concatenation of RT_NAME & "_" & RT_PART from the road centerline data
    rtNamepartStr = "LABEL"

    'Search tolerance in map units, ut from end points of eah route part
    calPtSearchTol = 1 'meters

    '***** END PARAMETERS
    '**************************

    Dim pMxDoc As IMxDocument
    Dim pMap As IMap
    Dim pCalibrationLayer As IFeatureLayer
    Dim pRoadsLayer As IFeatureLayer
    Dim pRoadsFC As IFeatureClass
    Dim pOutWS As IFeatureWorkspace
    Dim pOutFields As IFields
    Dim pOutFC As IFeatureClass
    Dim pGDS As IGeoDataset
    Dim pOutSR As ISpatialReference
    Dim dateStamp As String

    Set pMxDoc = ThisDocument
    Set pMap = pMxDoc.FocusMap
    Set pCalibrationLayer = pMap.Layer(calibrationLayerIndex)
    Set pGDS = pCalibrationLayer.FeatureClass
    Set pRoadsLayer = pMap.Layer(roadsLayerIndex)
    Set pRoadsFC = pRoadsLayer.FeatureClass
    Set pOutSR = pGDS.SpatialReference

    dateStamp = Format(Now, "yyyymmddhhmmss")
    Set pOutWS = openFGDBWS(outPath)
    Set pOutFields = createRouteFields(esriGeometryPolyline, pOutSR, True)
    Set pOutFC = createRouteFeatureClass(pOutWS, "Routes" & dateStamp, esriFTSimple, esriGeometryPolyline,
pOutFields)

    Dim csvRoutePartList As String
    Dim routePartList() As String
    Dim p As Long
    Dim routePartQStr As String
    Dim pRoutePartPolyline As IPolyline

    'Build Route Part List
    csvRoutePartList = getUniqueValues(pCalibrationLayer, rtNamepartStr)
    routePartList = Split(csvRoutePartList, ",")

    For p = 0 To UBound(routePartList)
        'Debug.Print routePartList(p)

        If routePartList(p) = "0302P_1" Then
            Debug.Print "here"
        End If

        routePartQStr = "DOT_RTNAME = '" & Left(routePartList(p), 5) & "' and DOT_RTPART = " _
                & Mid(routePartList(p), 7)
        Set pRoutePartPolyline = buildRoutePartPolyline(routePartQStr, pRoadsFC)

        If Not pRoutePartPolyline.IsEmpty Then
            Debug.Print routePartList(p) & ": " & pRoutePartPolyline.Length
```

```
'check for calibration end points
Dim pSpatialFilter As ISpatialFilter
Dim pEndPtTopOp As ITopologicalOperator
Dim pEndPtBuffer As IPolygon
Dim pCalPtFCursor As IFeatureCursor
Dim pCalPtFeature As IFeature
Dim pPoint1 As IPoint
Dim pPoint2 As IPoint
Dim point1M As Double
Dim point2M As Double
Dim ptempPoint As IPoint
Dim tempPointM As Double
Dim fromPointError As Boolean
Dim toPointError As Boolean
Dim pMSegmentation As IMSegmentation
Dim pOutFeature As IFeature
Dim pMAware As IMAware
Dim resultsCount As Long

'Find polyline FROMPOINT's corresponding calibration point
fromPointError = False
Set pEndPtTopOp = pRoutePartPolyline.FromPoint
Set pEndPtBuffer = pEndPtTopOp.Buffer(calPtSearchTol)
Set pSpatialFilter = New SpatialFilter
pSpatialFilter.WhereClause = rtNamepartStr & " = '" & routePartList(p) & "'"
Set pSpatialFilter.Geometry = pEndPtBuffer
pSpatialFilter.SpatialRel = esriSpatialRelContains

Set pCalPtFCursor = pCalibrationLayer.Search(pSpatialFilter, True)
Set pCalPtFeature = pCalPtFCursor.NextFeature
resultsCount = 0

Do Until pCalPtFeature Is Nothing
   resultsCount = resultsCount + 1
   If resultsCount = 1 Then
      Set pPoint1 = pCalPtFeature.ShapeCopy
      point1M = pCalPtFeature.value(pCalPtFeature.Fields.FindField(pCalibrationValueFieldName))
   ElseIf resultsCount > 1 Then
      Debug.Print "too many end calibration points for " & routePartList(p) & " composite polyline FROM POINT"
      fromPointError = True
   End If
   Set pCalPtFeature = pCalPtFCursor.NextFeature
Loop

If resultsCount = 0 Then
   Debug.Print "too few end calibration points for " & routePartList(p) & " composite polyline FROM POINT"
   fromPointError = True
End If


'Find polyline TOPOINT's corresponding calibration point
toPointError = False
Set pEndPtTopOp = pRoutePartPolyline.ToPoint
Set pEndPtBuffer = pEndPtTopOp.Buffer(calPtSearchTol)
Set pSpatialFilter = New SpatialFilter
pSpatialFilter.WhereClause = rtNamepartStr & " = '" & routePartList(p) & "'"
Set pSpatialFilter.Geometry = pEndPtBuffer
pSpatialFilter.SpatialRel = esriSpatialRelContains

Set pCalPtFCursor = pCalibrationLayer.Search(pSpatialFilter, True)
Set pCalPtFeature = pCalPtFCursor.NextFeature
resultsCount = 0

Do Until pCalPtFeature Is Nothing
```

```
            resultsCount = resultsCount + 1
            If resultsCount = 1 Then
               Set pPoint2 = pCalPtFeature.ShapeCopy
               point2M = pCalPtFeature.value(pCalPtFeature.Fields.FindField(pCalibrationValueFieldName))
            ElseIf resultsCount > 1 Then
               Debug.Print "too many end calibration points for " & routePartList(p) & " composite polyline TO POINT"
               toPointError = True
            End If
            Set pCalPtFeature = pCalPtFCursor.NextFeature
         Loop

         If resultsCount = 0 Then
            Debug.Print "too few end calibration points for " & routePartList(p) & " composite polyline TO POINT"
            toPointError = True
         End If

         If Not (fromPointError Or toPointError) Then
            'flip routepart polyline?
            If point1M > point2M Then
               Set ptempPoint = pPoint1
               Set pPoint1 = pPoint2
               Set pPoint2 = ptempPoint
               tempPointM = point1M
               point1M = point2M
               point2M = tempPointM
            End If

            If (Round(pRoutePartPolyline.FromPoint.x, calPtSearchTol) <> Round(pPoint1.x, calPtSearchTol)) Or _
              (Round(pRoutePartPolyline.FromPoint.y, calPtSearchTol) <> Round(pPoint1.y, calPtSearchTol)) Then
               pRoutePartPolyline.ReverseOrientation
            End If

            Set pMAware = pRoutePartPolyline
            pMAware.MAware = True
            Set pMSegmentation = pRoutePartPolyline
            pMSegmentation.SetAndInterpolateMsBetween point1M, point2M

         End If
         Set pOutFeature = pOutFC.CreateFeature
         With pOutFeature
            Set .Shape = pRoutePartPolyline
            .value(pOutFeature.Fields.FindField("LABEL")) = routePartList(p)
            .value(pOutFeature.Fields.FindField("RT_NAME")) = Left(routePartList(p), 4)
            .value(pOutFeature.Fields.FindField("RT_DIR")) = Mid(routePartList(p), 5, 1)
            .value(pOutFeature.Fields.FindField("RT_PART")) = Mid(routePartList(p), 7)
            .value(pOutFeature.Fields.FindField("EFF_DATE")) = Now
            .Store
         End With
      End If
   Next p



End Sub

Public Function createRouteFeatureClass(featWorkspace As IFeatureWorkspace, _
                                        Name As String, _
                                        featType As esriFeatureType, _
                                        geomType As esriGeometryType, _
                                        pFields As IFields _
                                        ) As IFeatureClass

  On Error GoTo EH
```

```
  Set createRouteFeatureClass = Nothing
  If featWorkspace Is Nothing Then Exit Function
  If Name = "" Then Exit Function

  Dim pCLSID As UID
  Set pCLSID = Nothing
  Set pCLSID = New UID

    '' determine the appropriate geometry type corresponding the the feature type
  Select Case featType
     Case esriFTSimple
       pCLSID.value = "esricore.Feature"
       If geomType = esriGeometryLine Then geomType = esriGeometryPolyline
     Case esriFTSimpleJunction
       geomType = esriGeometryPoint
       pCLSID.value = "esricore.SimpleJunctionFeature"
     Case esriFTComplexJunction
       pCLSID.value = "esricore.ComplexJunctionFeature"
     Case esriFTSimpleEdge
       geomType = esriGeometryPolyline
       pCLSID.value = "esricore.SimpleEdgeFeature"
     Case esriFTComplexEdge
       geomType = esriGeometryPolyline
       pCLSID.value = "esricore.ComplexEdgeFeature"
     Case esriFTAnnotation
       Exit Function
  End Select


  ' establish the class extension
  Dim pCLSEXT As UID
  Set pCLSEXT = Nothing

  ' locate the shape field
  Dim strShapeFld As String
  Dim j As Integer
  For j = 0 To pFields.FieldCount - 1
    If pFields.Field(j).Type = esriFieldTypeGeometry Then
      strShapeFld = pFields.Field(j).Name
    End If
  Next

  Set createRouteFeatureClass = featWorkspace.CreateFeatureClass(Name, pFields, pCLSID, _
                        pCLSEXT, featType, strShapeFld, "")

  Exit Function
EH:
    MsgBox Err.Description, vbInformation, "createWorkspaceFeatureClass"
End Function
Public Function createRouteFields(geomType As Long, pSR As ISpatialReference, _
                             hasM As Boolean) As IFields
    Dim pField As IField
    Dim pFields As IFields
    Dim pFieldEdit As IFieldEdit
    Dim pFieldsEdit As IFieldsEdit
    Dim hasmcoord As Boolean


    'Create new Fields collection
    Set pFields = New Fields
    Set pFieldsEdit = pFields
    'pFieldsEdit.FieldCount = 1

    ''
    '' create the geometry field
```

```
''
Dim pGeomDef As IGeometryDef
Set pGeomDef = New GeometryDef
Dim pGeomDefEdit As IGeometryDefEdit
Set pGeomDefEdit = pGeomDef

' assign the spatial reference
'Dim pSR As ISpatialReference
If pSR Is Nothing Then
    Set pSR = New UnknownCoordinateSystem
    pSR.SetFalseOriginAndUnits 0, 0, 100
End If

pSR.SetMFalseOriginAndUnits -100000, 1000

If Not hasM Then
    hasmcoord = False
Else
    hasmcoord = True
End If

'' assign the geometry definiton properties.
With pGeomDefEdit
  .GeometryType = geomType
  .GridCount = 3
  .GridSize(0) = 1000
  .GridSize(1) = 10000
  .GridSize(2) = 100000
  .AvgNumPoints = 200
  .hasM = hasmcoord
  .HasZ = False
  Set .SpatialReference = pSR
End With

Set pField = New Field
Set pFieldEdit = pField

pFieldEdit.Name = "Shape"
pFieldEdit.Type = esriFieldTypeGeometry
Set pFieldEdit.GeometryDef = pGeomDef
pFieldsEdit.AddField pField


'Create Object ID Field
Set pField = New Field
Set pFieldEdit = pField

With pFieldEdit
    .Name = "OBJECTID"
    .AliasName = "FID"
    .Type = esriFieldTypeOID
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Length = 10
    .Name = "LABEL"
    .Type = esriFieldTypeString
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
```

```
With pFieldEdit
    .Length = 4
    .Name = "RT_NAME"
    .Type = esriFieldTypeString
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Length = 1
    .Name = "RT_DIR"
    .Type = esriFieldTypeString
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Name = "RT_PART"
    .Type = esriFieldTypeSmallInteger
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Name = "RT_DIR_ID"
    .Type = esriFieldTypeInteger
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Name = "EFF_DATE"
    .Type = esriFieldTypeDate
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Name = "DEP_DATE"
    .Type = esriFieldTypeDate
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Length = 100
    .Name = "EFF_NOTES"
    .Type = esriFieldTypeString
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Length = 100
    .Name = "DEP_NOTES"
    .Type = esriFieldTypeString
End With
pFieldsEdit.AddField pField
```

```
    Set pFields = pFieldsEdit

    Set createRouteFields = pFields

End Function

Public Function openFGDBWS(inPath As String) As IFeatureWorkspace
        Dim pFGDBWSFactory As IWorkspaceFactory
        Set pFGDBWSFactory = New esriDataSourcesGDB.FileGDBWorkspaceFactory
        Set openFGDBWS = pFGDBWSFactory.OpenFromFile(inPath, 0)
End Function

Public Function getUniqueValues(inFL As IFeatureLayer, sFieldName As String) As String
      Dim pData As esriGeoDatabase.IDataStatistics
      Dim pCursor As esriGeoDatabase.ICursor
      Dim pStatResults As esriSystem.IStatisticsResults

      Set pCursor = inFL.Search(Nothing, False)

      Set pData = New esriGeoDatabase.DataStatistics
      pData.Field = sFieldName
      Set pData.Cursor = pCursor

      Dim pEnumVar As esriSystem.IEnumVariantSimple, value As Variant
      Dim iCnt As Integer
      iCnt = 0
      Set pEnumVar = pData.UniqueValues
      value = pEnumVar.Next

      Do Until IsEmpty(value)
        If iCnt = 0 Then
           getUniqueValues = value
        Else
           getUniqueValues = getUniqueValues + "," & value
        End If
        value = pEnumVar.Next
        iCnt = iCnt + 1
      Loop
End Function


Public Function buildRoutePartPolyline(inRoutePartQueryString As String, inRoadFC As IFeatureClass) As IPolyline

    Dim pFCursor As IFeatureCursor
    Dim pfeature As IFeature
    Dim pQF As IQueryFilter
    Dim pInGC As IGeometryCollection
    Dim pOutGC As IGeometryCollection
    Dim pGeometry As IGeometry
    Dim g As Long

    Set pQF = New QueryFilter
    pQF.WhereClause = inRoutePartQueryString
    Set pFCursor = inRoadFC.Search(pQF, True)
    Set pfeature = pFCursor.NextFeature
    Set pOutGC = New Polyline

    Do Until pfeature Is Nothing

        Set pGeometry = pfeature.ShapeCopy
        Set pInGC = pGeometry
        For g = 0 To pInGC.GeometryCount - 1
           pOutGC.AddGeometry pInGC.Geometry(g)
        Next g
```

```
    Set pfeature = pFCursor.NextFeature
  Loop

  pOutGC.GeometriesChanged
  Set buildRoutePartPolyline = pOutGC
  buildRoutePartPolyline.SimplifyNetwork

End Function
```